IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Amith Vardhan Reddy
Surasani
May 17, 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- **Summary of methodologies:**

1. Data Collection using following methods:

- API
- Web Scraping

2. Data Wrangling
3. Exploratory Data Analysis:

- Structured Query Language (SQL)
- Data Visualization (Matplotlib, Pandas)

4. Interactive Visual Analytics using Folium
5. Dashboard using Plotly Dash
6. Predictive Analysis with Machine Learning

- **Summary of all results:**

1. Results obtained from Exploratory Data Analysis
2. Screenshots for Interactive Visual Analytics using Folium
3. Results obtained from Predictive Analytics using Machine Learning techniques

Introduction

- The Space Exploration Technologies Corporation, also called as SpaceX is an organization which deals with design, simulation, assembling, testing and operation of spacecrafts and facilitating satellite communication.
- Its main motive is to make spacecrafts at a lower cost for colonizing Mars.
- It was established in the year 2002 by one of the famous business tycoons, Elon Musk.
- The organization takes care of Falcon 9, Falcon Heavy, and Starship launch vehicles and rocket engines and Starlink satellite.
- The organization is well known for its high-tech launch vehicles with minimum investment. The best example is Falcon 9, whose investment is 62 million USD. The same vehicle would have taken 165 million USD for other space agencies.
- As a data scientist working in SpaceX, the main intention is to predict the insights for landing of the spacecrafts so that the best quality vehicle can be made at a very low cost.
- Problems for finding answers:

1) Determination of the criteria which impact the landing insights of spacecrafts
2) Estimation of the relationship between the criteria and the landing insights
3) Finding the corresponding measures of the criteria for most effective landing insights.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected from SpaceX REST API and scrapping the Wikipedia page about SpaceX.
- Perform data wrangling
  - Used one-hot encoding technique, i.e., getting dummy variables for categorical variables
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
- Data collection is the act of gathering data on particular criteria from various sources available for effective analysis and machine learning. In this project, the data is collected using two methods:

1) REST API: The content from the REST API is taken using the get() function and stored as response object. Using this object, the collected data is transformed into a JSON file, which in turn is transformed to a pandas DataFrame using json_normalize() method.
2) Web Scrapping: The content from the REST API is taken using the get() function and stored as text. Then, the BeautifulSoup object is used to get all the contents from the website. All the launch records are taken as HTML table, which in turn is converted into pandas dataframe.

Data Collection – SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

```
data=pd.json_normalize(response.json())
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Get request for rocket launch data using API

↓

Use json_normalize method to convert json result to dataframe

↓

Performed data cleaning and filling the missing value

Data Collection - Scraping

```
# use requests.get() method with the provided static_url
# assign the response to a object
data=requests.get(static_url).text
```

```
soup=BeautifulSoup(data,'html.parser')
```

Get request for Falcon9 Launch Wikipedia page

↓

Create a BeautifulSoup object from the HTML response

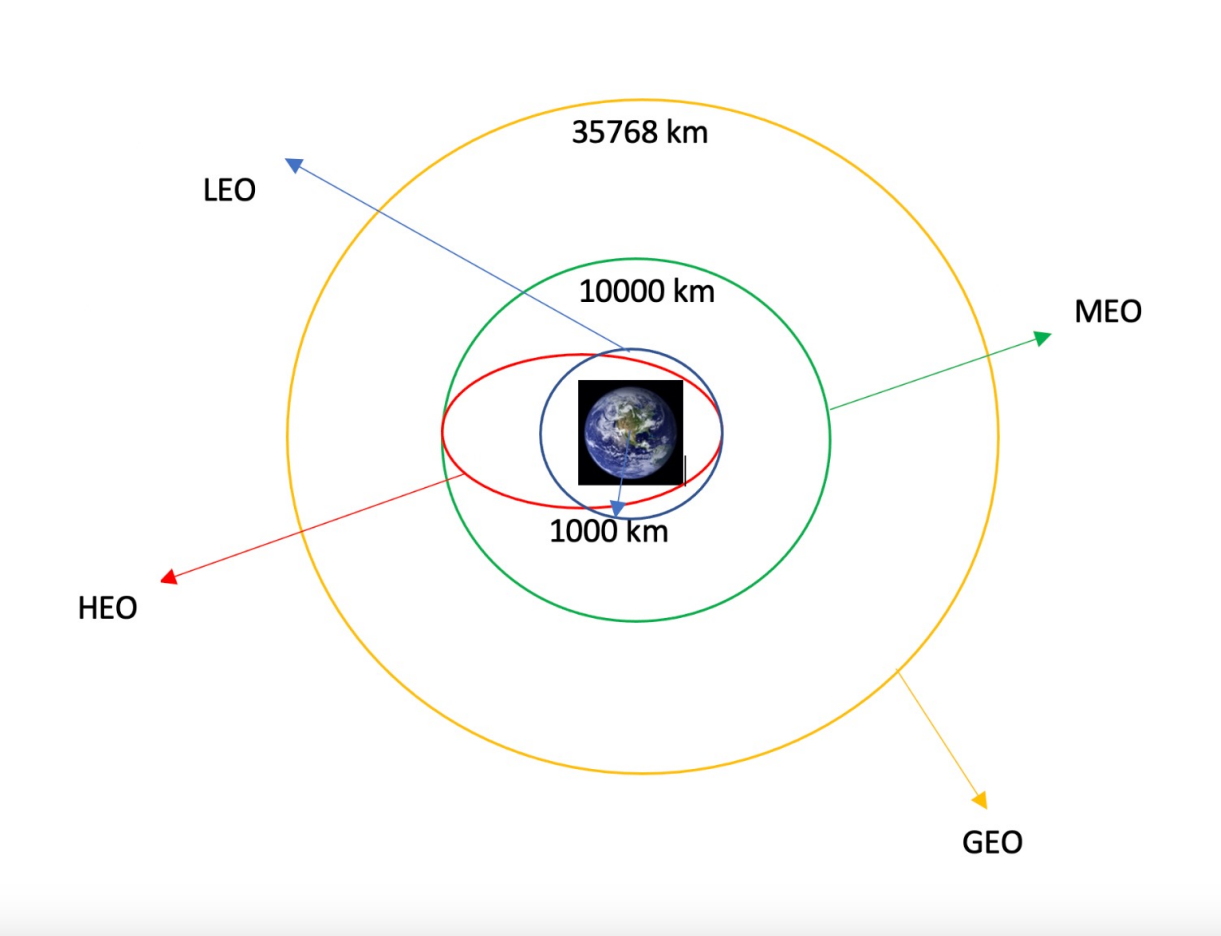
↓

Extract all column/variable names from the HTML header

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            launch_dict['Flight No.'].append(flight_number)
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            #print(time)
            print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv)
            print(bv)
```

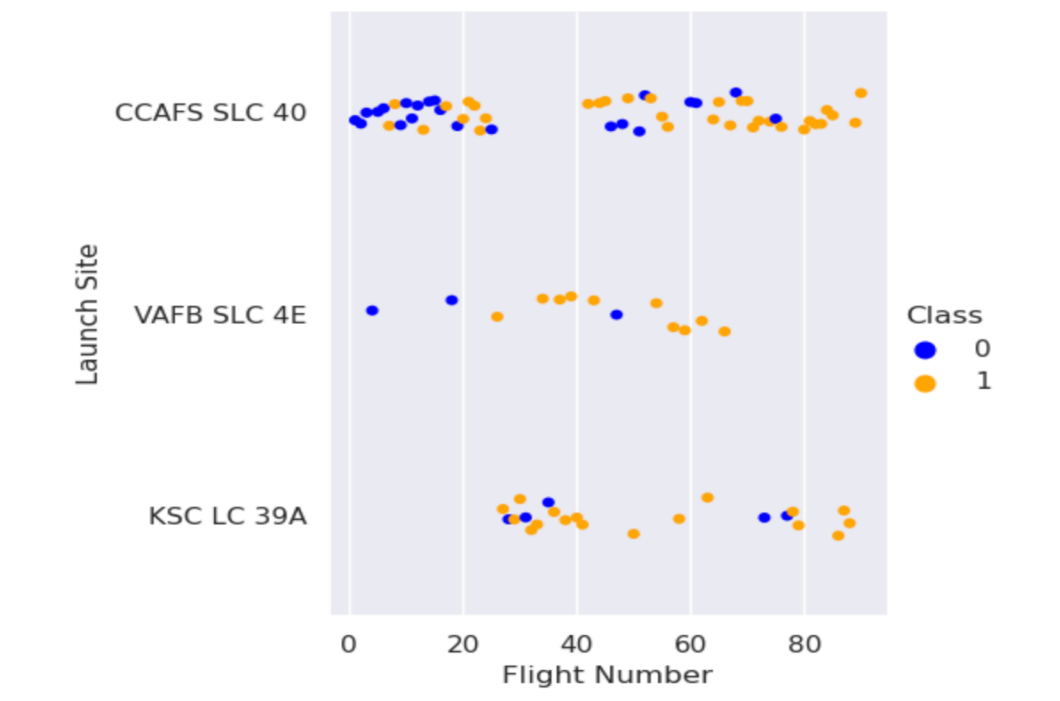
```
# Launch Site
# TODO: Append the bv into launch_dict with key `Launch Site`
launch_site = row[2].a.string
launch_dict['Launch site'].append(launch_site)
#print(launch_site)
print(launch_site)

# Payload
# TODO: Append the payload into launch_dict with key `Payload`
payload = row[3].a.string
launch_dict['Payload'].append(payload)
#print(payload)
print(payload)

# Payload Mass
# TODO: Append the payload_mass into launch_dict with key `Payload mass`
payload_mass = get_mass(row[4])
launch_dict['Payload mass'].append(payload_mass)
#print(payload)
print(payload)

# Orbit
# TODO: Append the orbit into launch_dict with key `Orbit`
orbit = row[5].a.string
launch_dict['Orbit'].append(orbit)
#print(orbit)
print(orbit)

# Customer
# TODO: Append the customer into launch_dict with key `Customer`
customer = row[6].a
launch_dict['Customer'].append(customer)
#print(customer)
print(customer)

# Launch outcome
# TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
launch_outcome = list(row[7].strings)[0]
launch_dict['Launch outcome'].append(launch_outcome)
#print(launch_outcome)
print(launch_outcome)

# Booster landing
# TODO: Append the launch_outcome into launch_dict with key `Booster landing`
booster_landing = landing_status(row[8])
launch_dict['Booster landing'].append(booster_landing)
#print(booster_landing)
print(booster_landing)
```

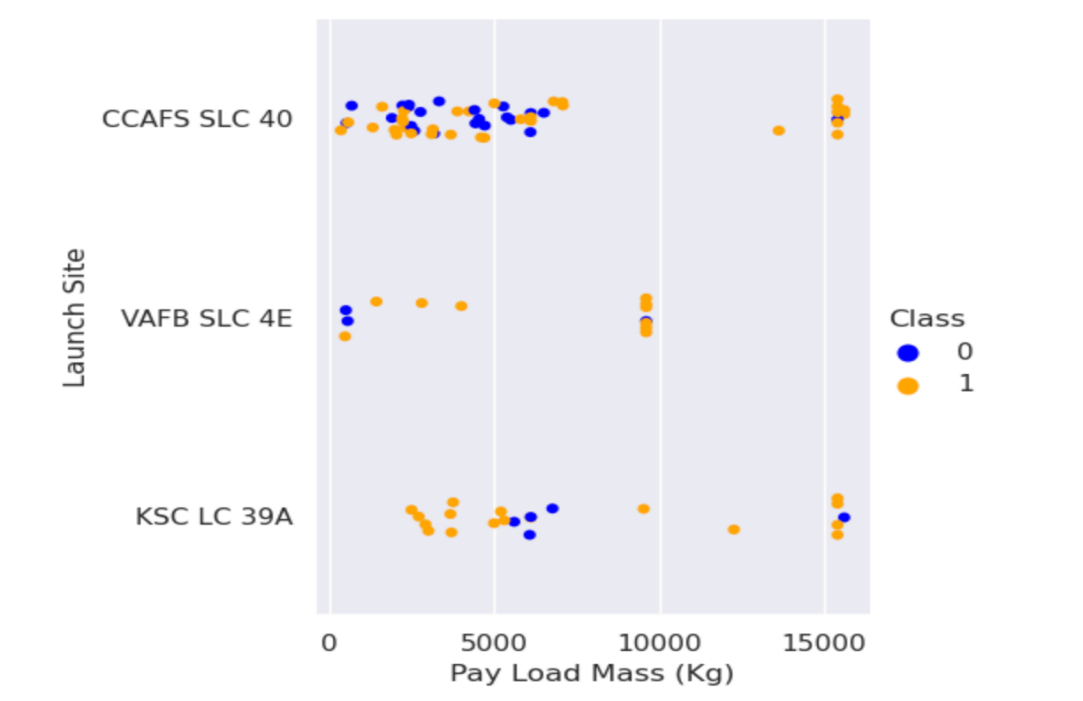
Data Wrangling



1. The complicated datasets deserve to be cleaned for effective Exploratory Data Analysis (EDA) and Machine Learning. This process is called data wrangling.
2. The number of launches will be computer per launch site, followed by the number of mission outcomes for each type of orbit and occurrence of each outcome.
3. Labels will be created for landing results label from the outcome variable, which simplifies further data analysis, visualization, and Machine Learning.
4. In the end, the result will be exported to a CSV file.

EDA with Data Visualization

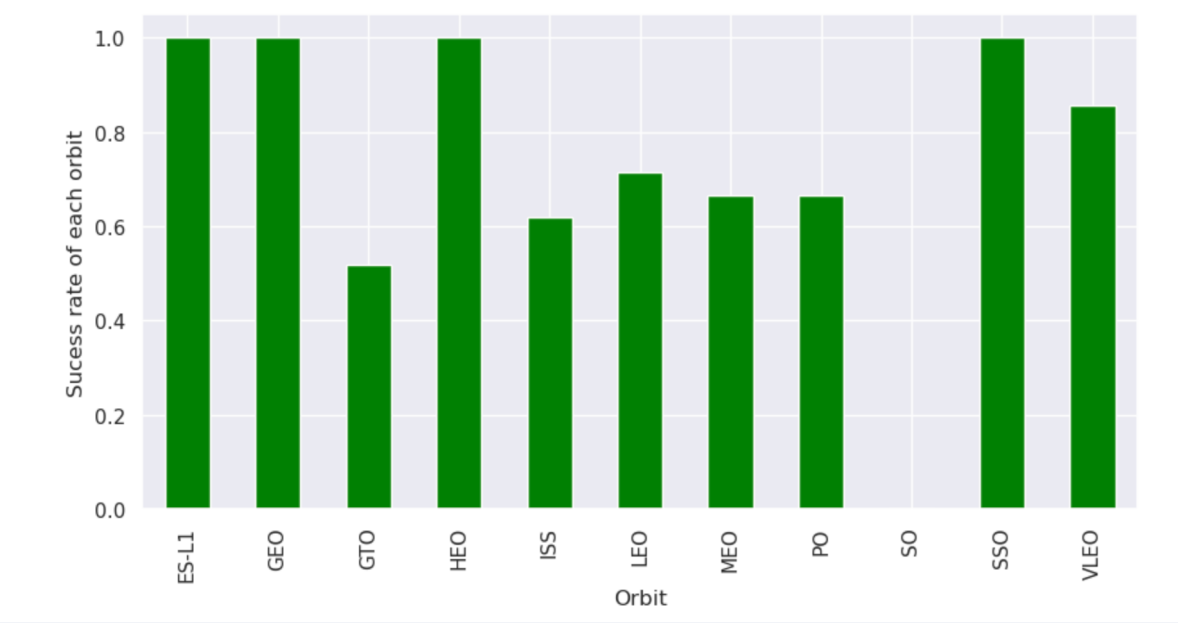


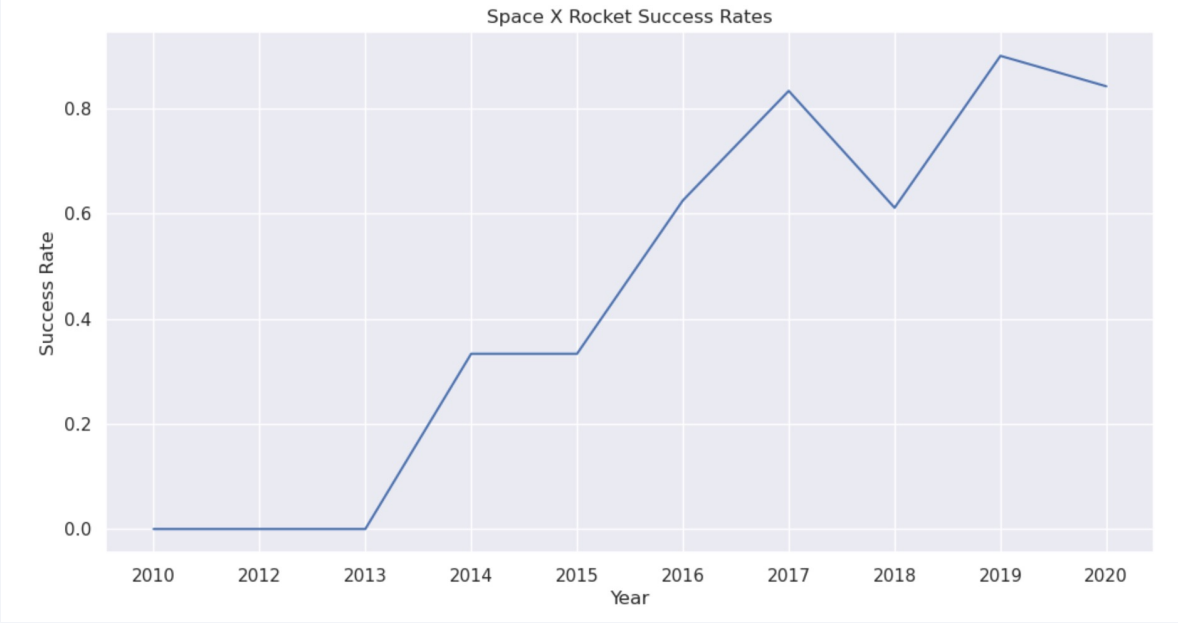


Scatter graphs are constructed to depict how the following variables vary with each other:

- Payload Mass and Flight Number.
- Flight Number and Launch Site.
- Payload Mass and Launch Site.
- Flight Number and Orbit Type.
- Payload Mass and Orbit Type

EDA with Data Visualization





- Bar Graphs and Line Graphs are used to go deep into further analysis.
- In this case, bar graph is used to plot the success probabilities for the orbits.
- The Line Graph is used to display the success rate as time progresses.
- Feature Engineering is used for categorical variables in forecasting the successes of spacecrafts.

EDA with SQL

- We used numerous queries on the dataset for exploring and analyzing the data. The following are some of them:
- Names of launch sites.
- 5 records with launch sites starting with ‘CCA’.
- Total payload mass carried by booster launched by NASA (CRS).
- Average payload mass carried by booster version F9 v1.1.
- Date of the achievement of the first successful landing result in ground pad.
- Names of the boosters which are successful in drone ship and have payload mass between 4000 and 6000.
- Total number of successful and failure results.

Build an Interactive Map with Folium

- The latitude and longitude coordinates are taken at each launch site and given a specific name.
- Haversine's formula is used to find the distance between launch sites and various landmarks.

Build a Dashboard with Plotly Dash

- A dashboard is developed using Plotly dash which allows users to do anything with the data.
- Pie charts are constructed which display the total launches at specific launch sites.
- The scatter graph depicts how Outcome and Payload Mass (Kg) vary with each other.

Predictive Analysis (Classification)

1) Building the Model

- Load the dataset
- Transform the data
- Split the data into training and testing datasets
- Deciding the Machine Learning to be used
- Set the parameters and algorithms to GridSearchCV and fit it to dataset.

2) Evaluating the Model

- Check the accuracy for each model
- Get tuned hyperparameters for each type of algorithms.
- Plot the confusion matrix

3) Improving the Model

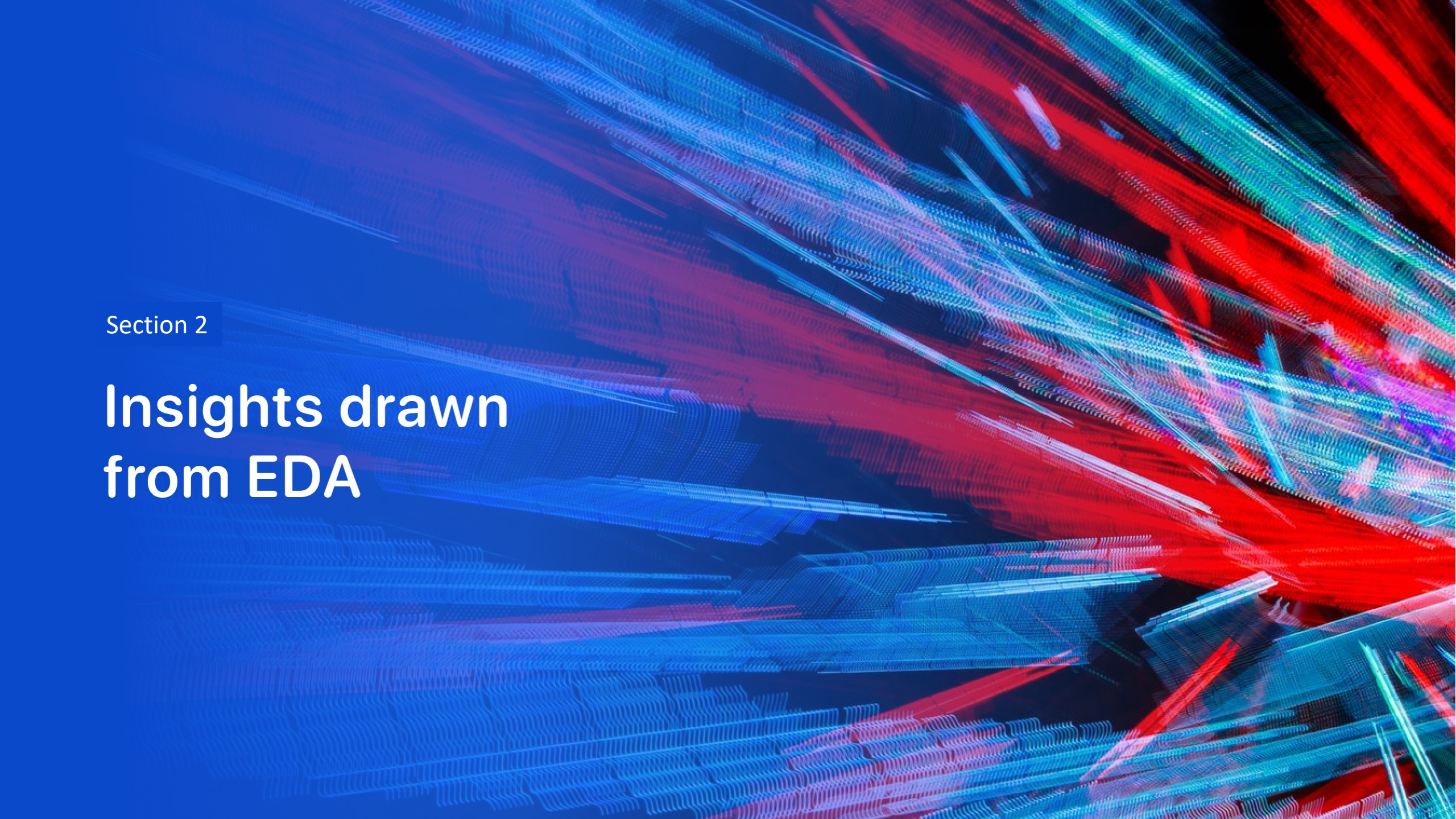
- Use Feature Engineering and Algorithm Tuning

4) Find the Best Model

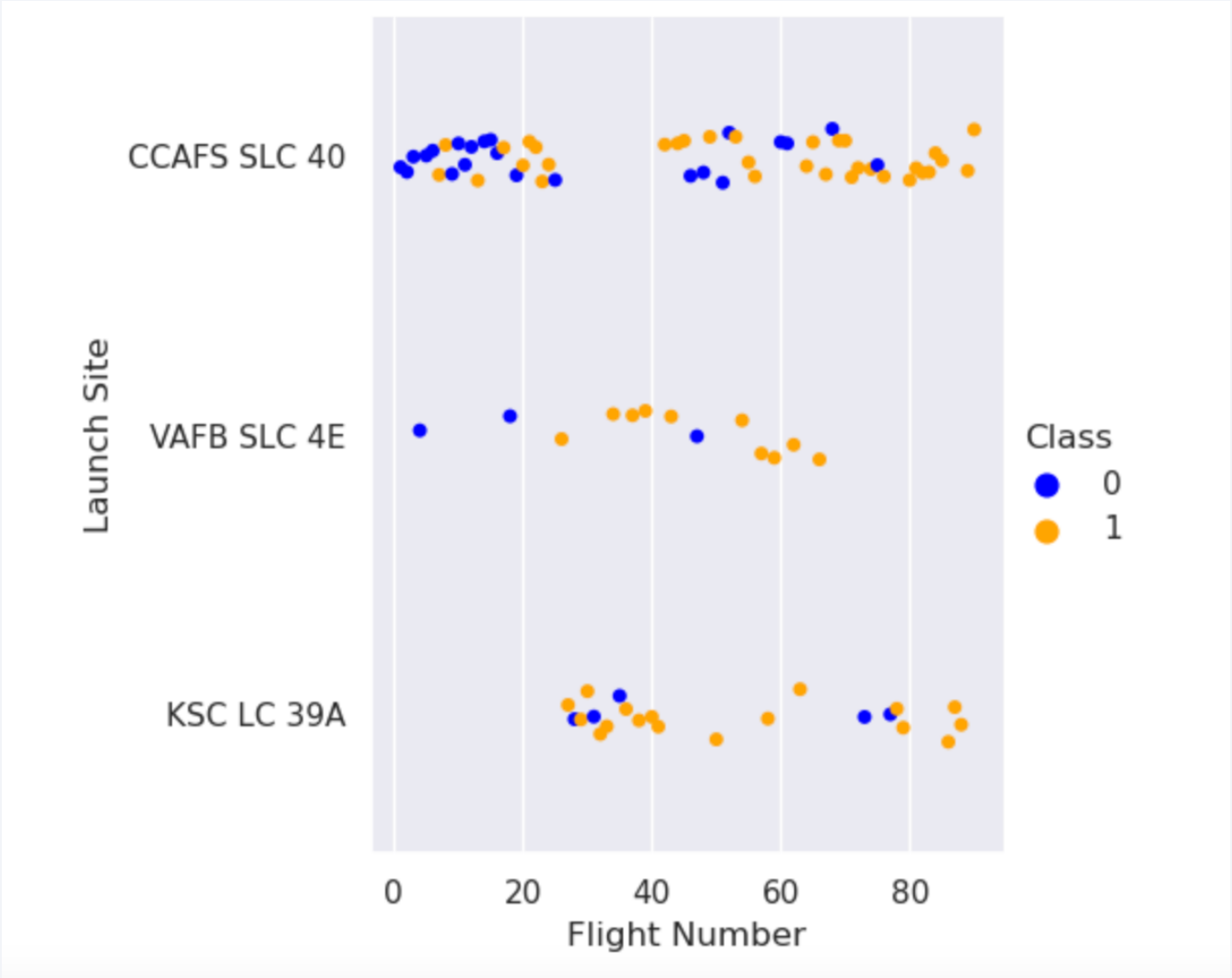
- Choose the model with the best accuracy score

Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

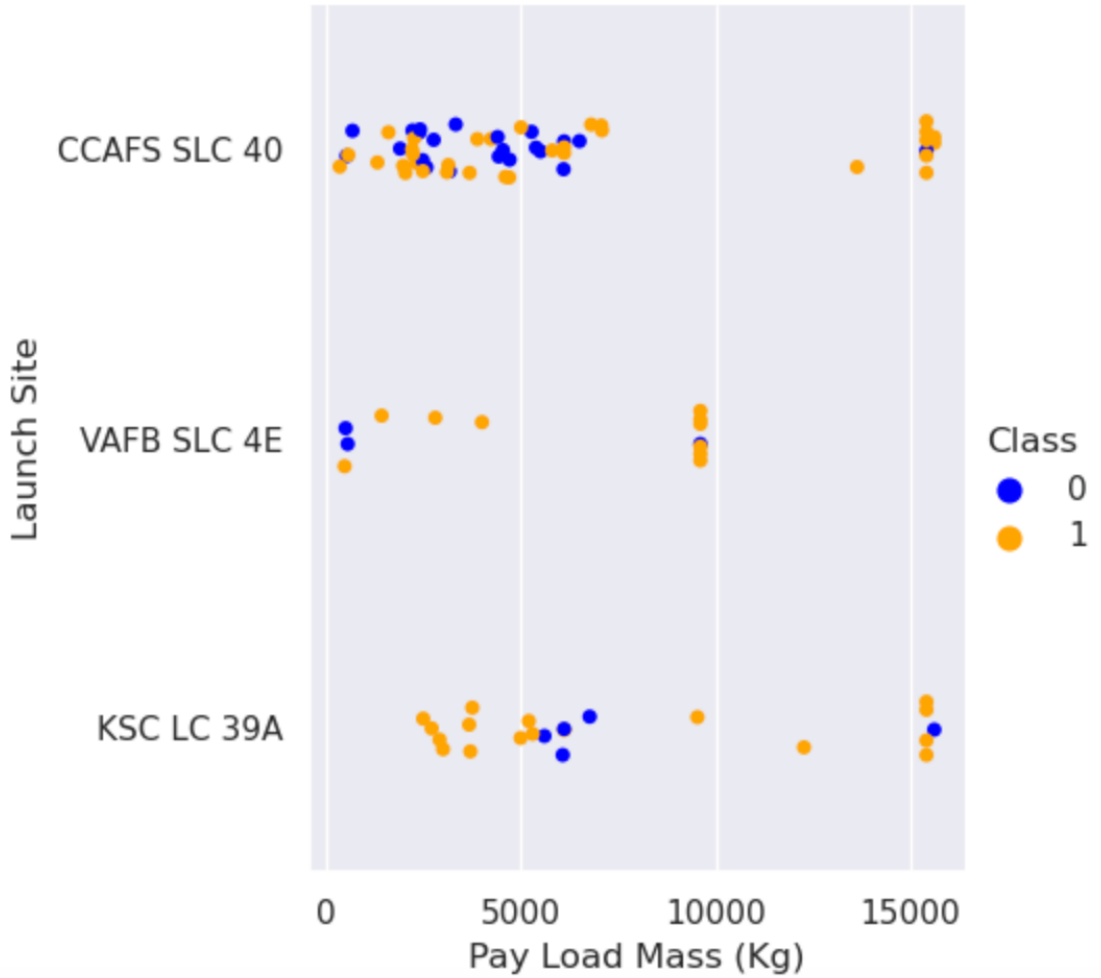

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



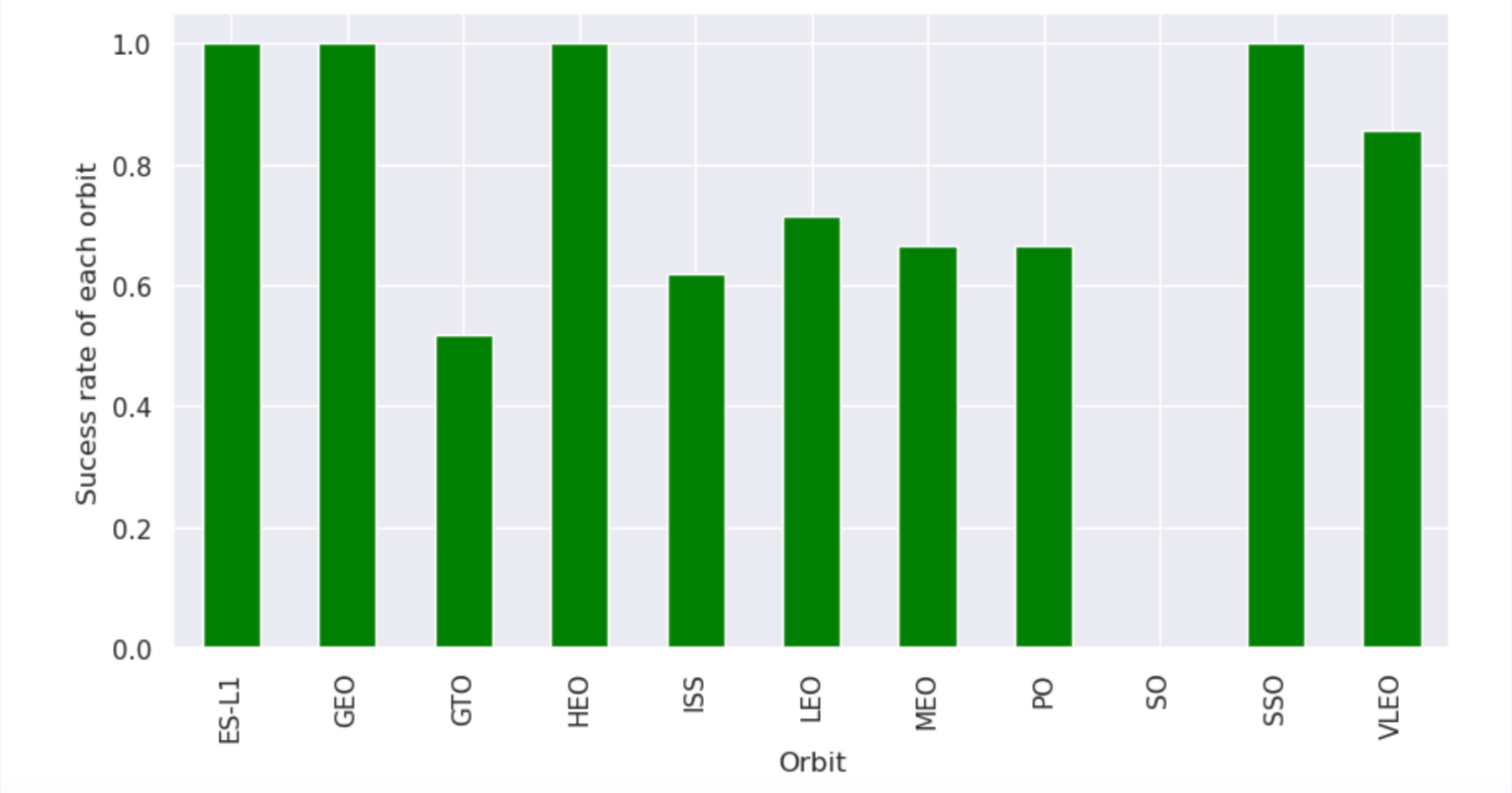
- This figure depicts that number of flights from the launch point is directly proportional to the success rate.
- CCAFS SLC40 is the only exception.

Payload vs. Launch Site

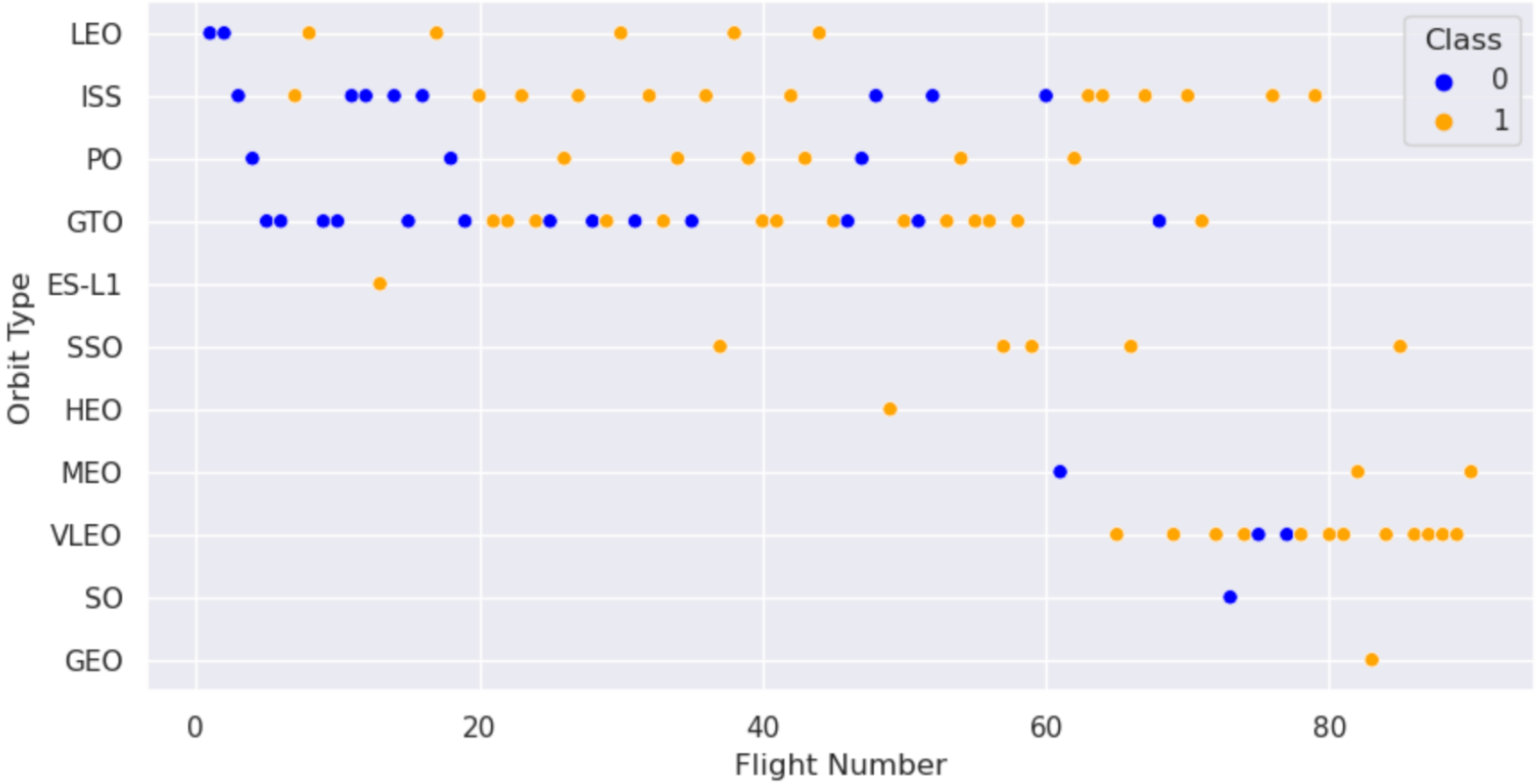


- This plot shows that pay load mass results in rapid increase of the success probability. This happens only if payload mass is >7000 kg.
- For the rest, there is no clear relation.

Success Rate vs. Orbit Type

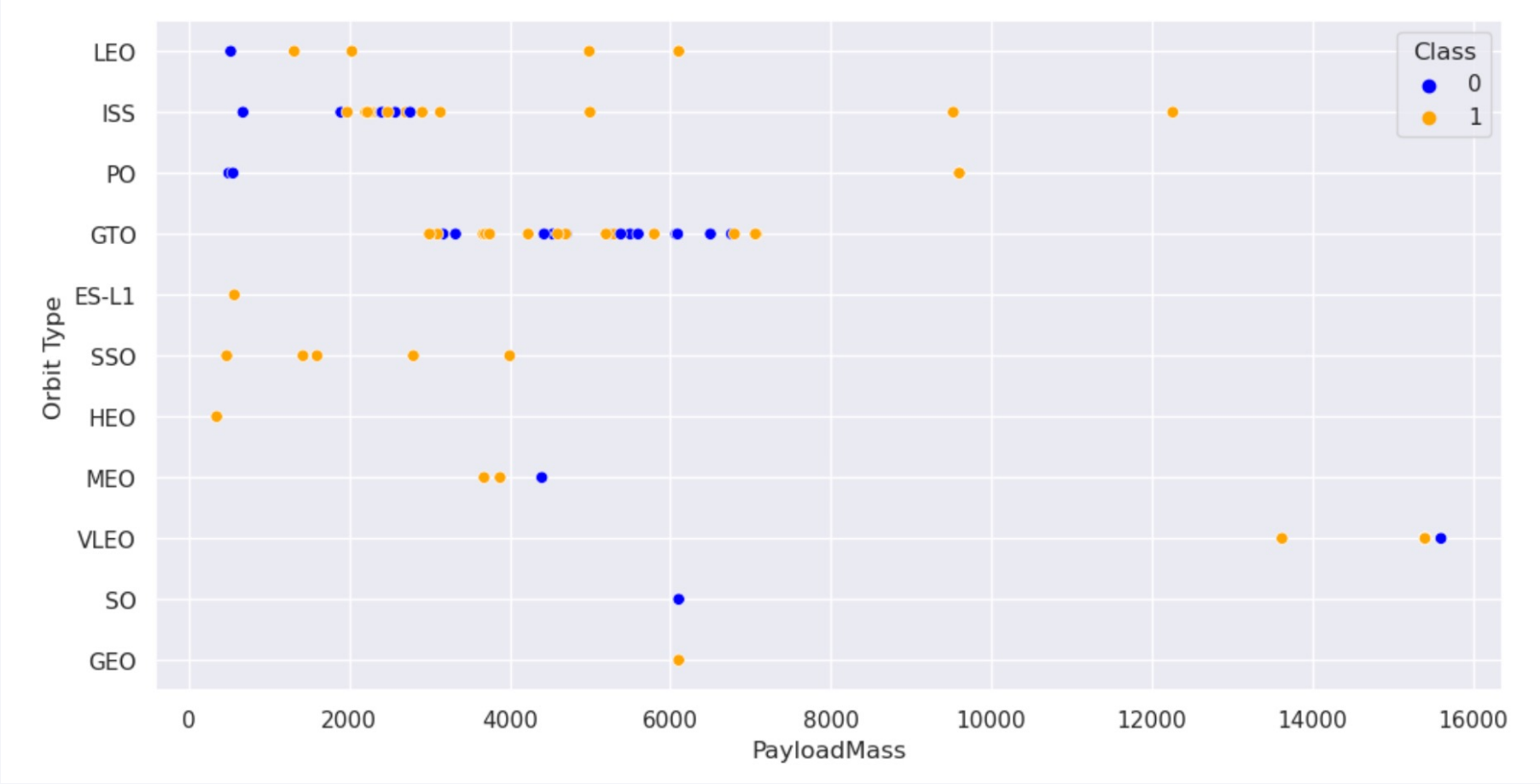


Flight Number vs. Orbit Type



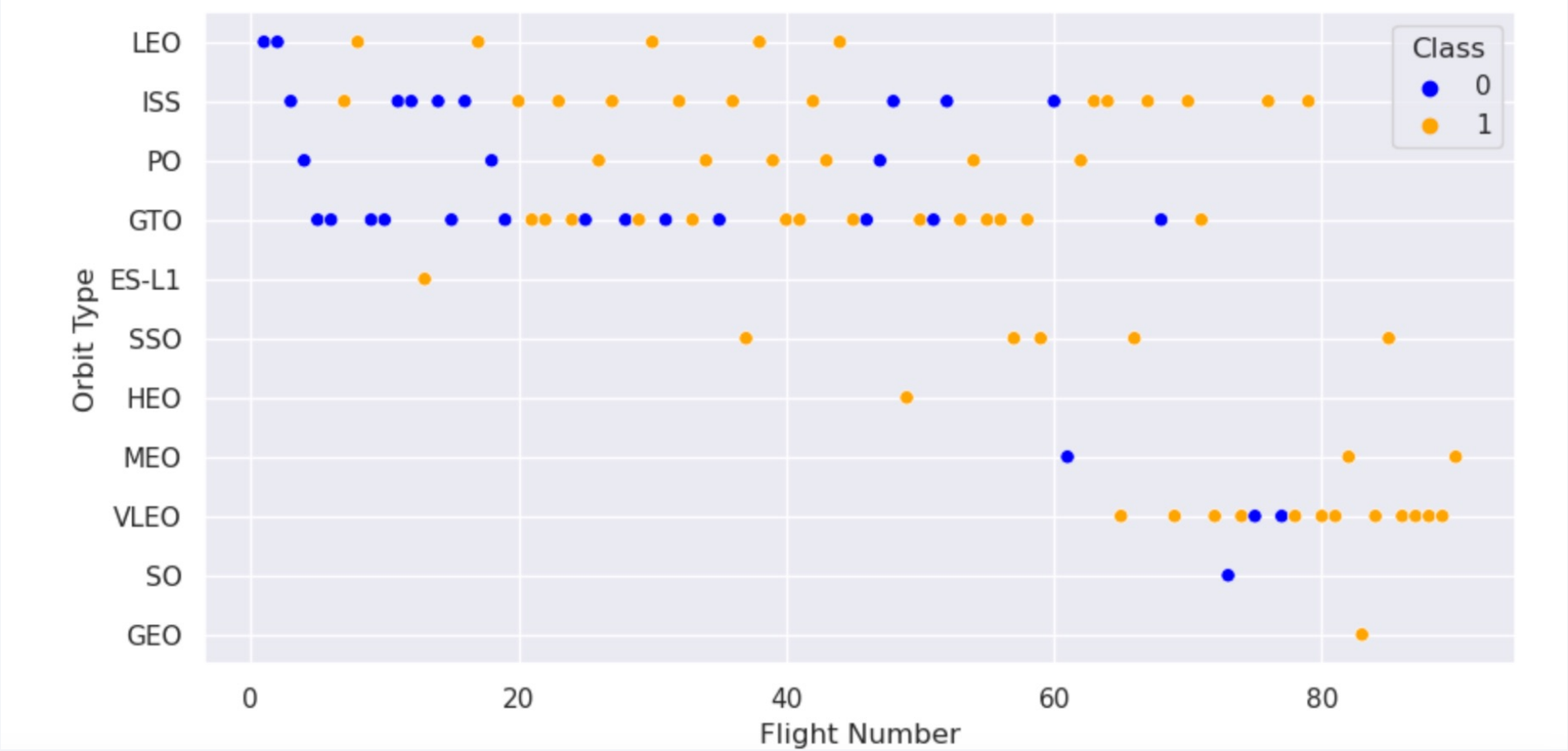
- Heavier payload has good impact on LEO, ISS and PO orbit.
- However, it has negative impact on MEO and VLEO orbit.
- GTO orbit has no relation with the payload.

Payload vs. Orbit Type



- The graph depicts the direct proportion of flight number on each orbits, with the success rate except for GTO orbit, for which there is no relationship.

Launch Success Yearly Trend



This figure clearly depicts an increasing trend from the year 2013 until 2020.

All Launch Site Names

- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.

In [4]:
```
%%sql
select distinct LAUNCH_SITE as "Launch_Sites" from SPACEX
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[4]:

Launch_Sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Launch Site Names Begin with 'CCA'

Task 2

Display 5 records where launch sites begin with the string 'CCA'

In [5]:
```
%%sql
select LAUNCH_SITE from SPACEX where LAUNCH_SITE like 'CCA%' limit 5;
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[5]:

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [8]:
```
%%sql
select sum(PAYLOAD_MASS_KG_) from SPACEX where CUSTOMER = 'NASA(CRS)';
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[8]:

1

None

Average Payload Mass by F9 v1.1

Task 4

Display average payload mass carried by booster version F9 v1.1

In [11]:
```
%%sql
select avg(PAYLOAD_MASS_KG_) from SPACEX where BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[11]:

1
2928

First Successful Ground Landing Date

We use the min() function to find the result We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

In [12]:
```
%%sql
select min(DATE) AS "First Successful Landing" FROM SPACEX WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[12]:

First Successful Landing
2015-12-22

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [15]:
```
%%sql
select count(MISSION_OUTCOME) as "Succesful Mission" from SPACEX where MISSION_OUTCOME like 'Success%'
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[15]:

Succesful Mission
100

In [16]:
```
%%sql
select count(MISSION_OUTCOME) as "Failure Mission" from SPACEX where MISSION_OUTCOME like 'Fail%'
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[16]:

Failure Mission
1

Boosters Carried Maximum Payload

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [21]:
```
%%sql
select distinct BOOSTER_VERSION as "Booster Versions which carried the Maximum Payload Mass" from SPACEX
where PAYLOAD_MASS_KG_ =(select max(PAYLOAD_MASS_KG_) from SPACEX);
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[21]:

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [22]:
```
%%sql
select BOOSTER_VERSION, LAUNCH_SITE from SPACEX where DATE like '2015-%' and
LANDING_OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

Out[22]:

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

A combination of the WHERE clause, LIKE, AND, and BETWEEN conditions is used to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

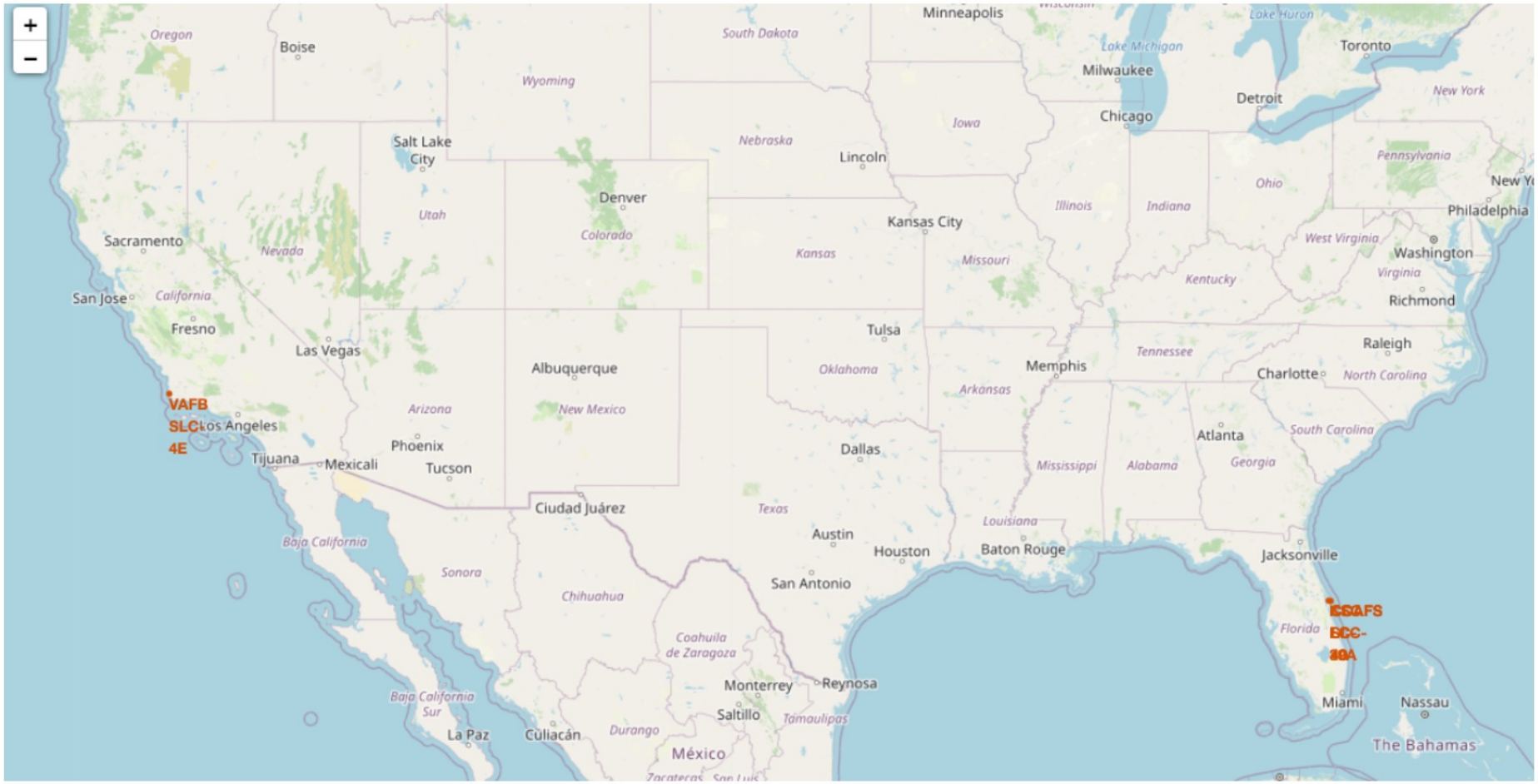
In [27]:

```
%%sql
select LANDING__OUTCOME as "Landing Outcome", count(LANDING_OUTCOME) as "Total Count" from SPACEX
where DATE between '2010-06-04' and '2017-03-20'
group by LANDING_OUTCOME
order by count(LANDING_OUTCOME) desc;
```

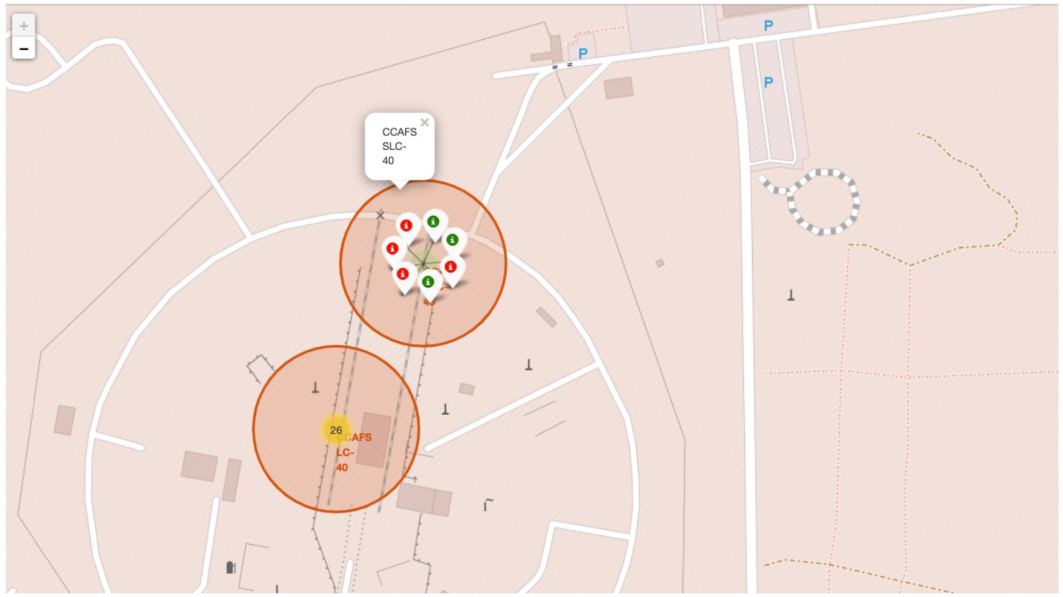
```
 * ibm_db_sa://jjz06171:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:307
56/BLUDB
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: SQLNumResultCols failed: [IBM][CLI Driver][DB2/LINUXX86
64] SQL0206N  "LANDING__OUTCOME" is not valid in the context where it is used.  SQLSTATE=42703 SQLCODE=-206
[SQL: select LANDING__OUTCOME as "Landing Outcome", count(LANDING_OUTCOME) as "Total Count" from SPACEX
where DATE between '2010-06-04' and '2017-03-20'
group by LANDING_OUTCOME
order by count(LANDING_OUTCOME) desc;]
(Background on this error at: http://sqlalche.me/e/13/f405)
```

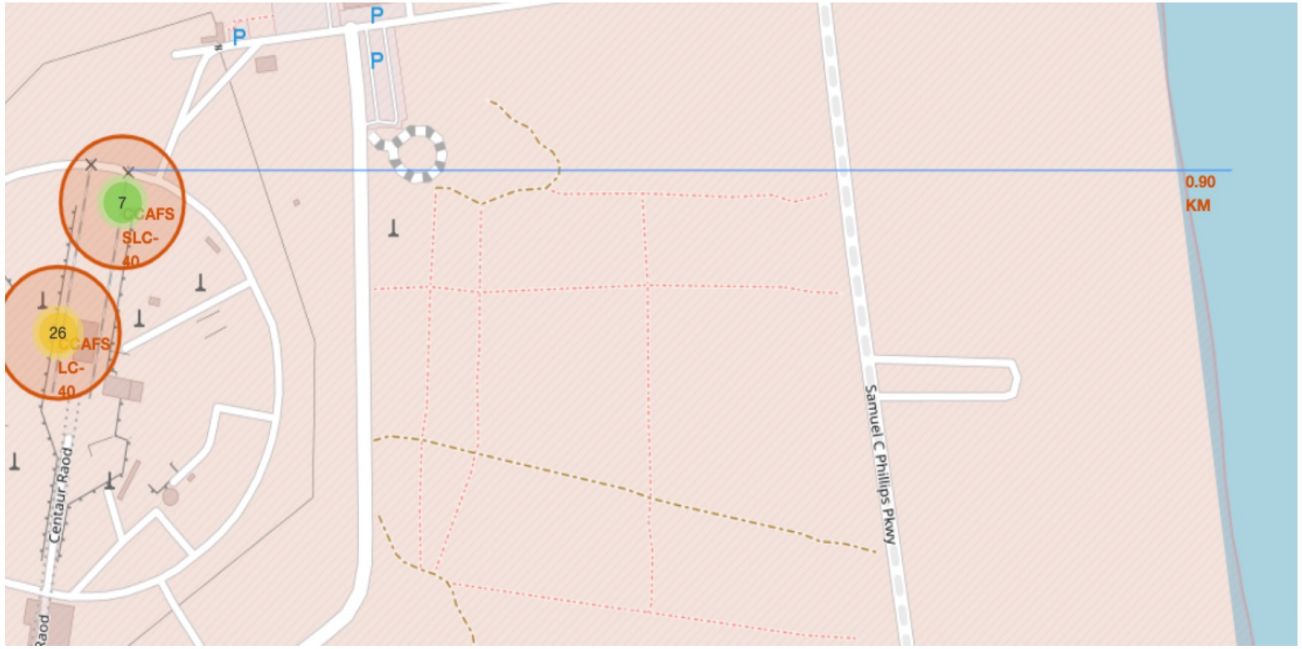

Section 3
Launch Sites
Proximities Analysis

Location of all the Launch Sites

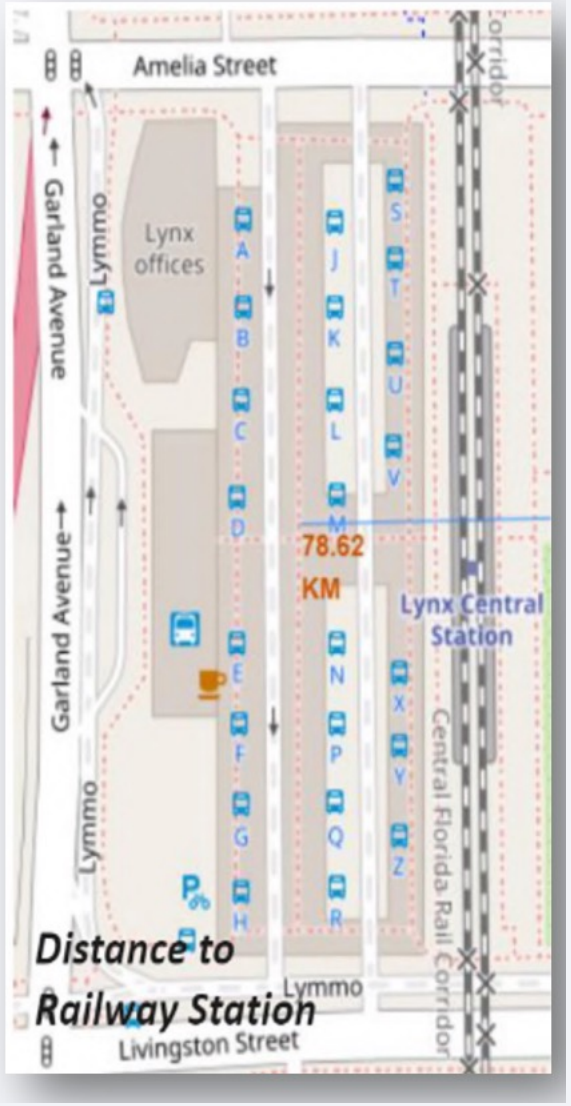


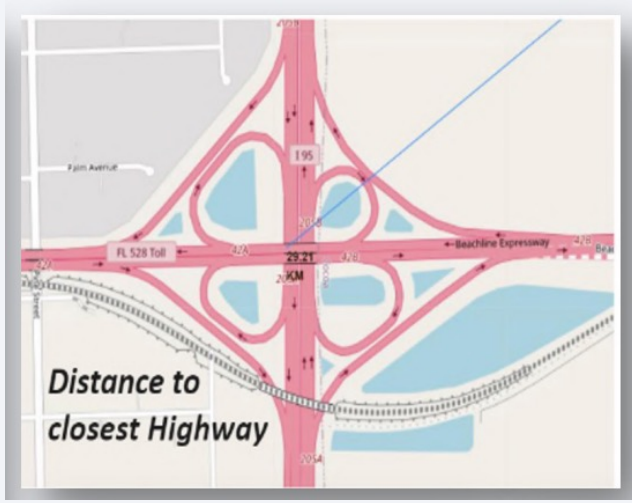
Markers showing launch sites with colour labels

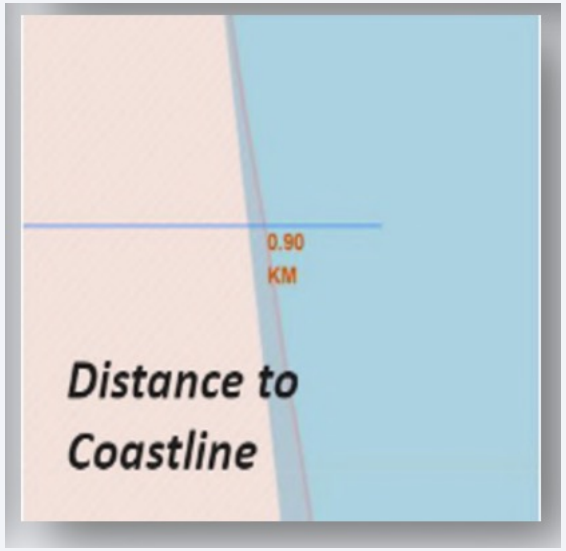


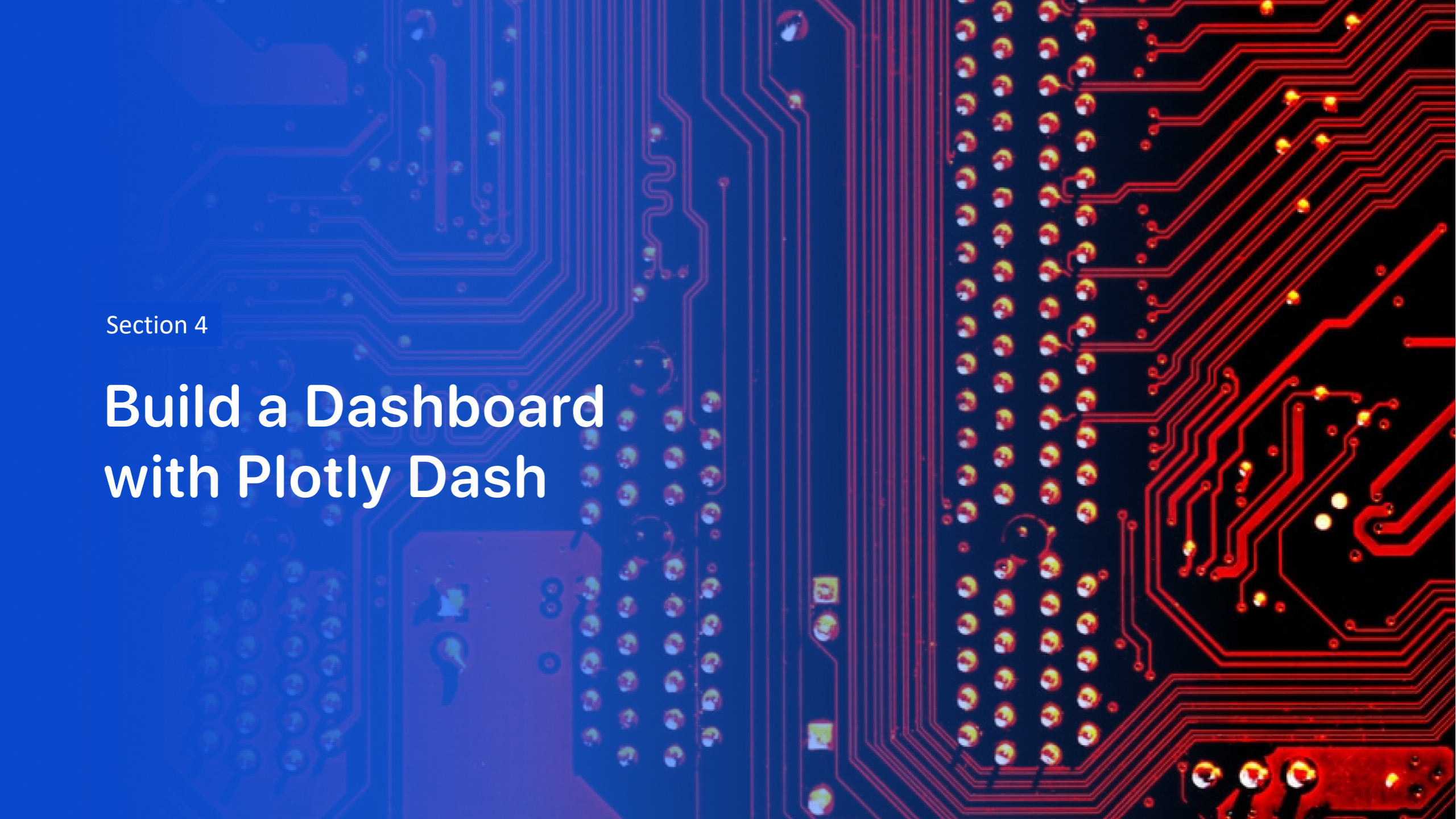


Launch Sites Distance to Landmarks

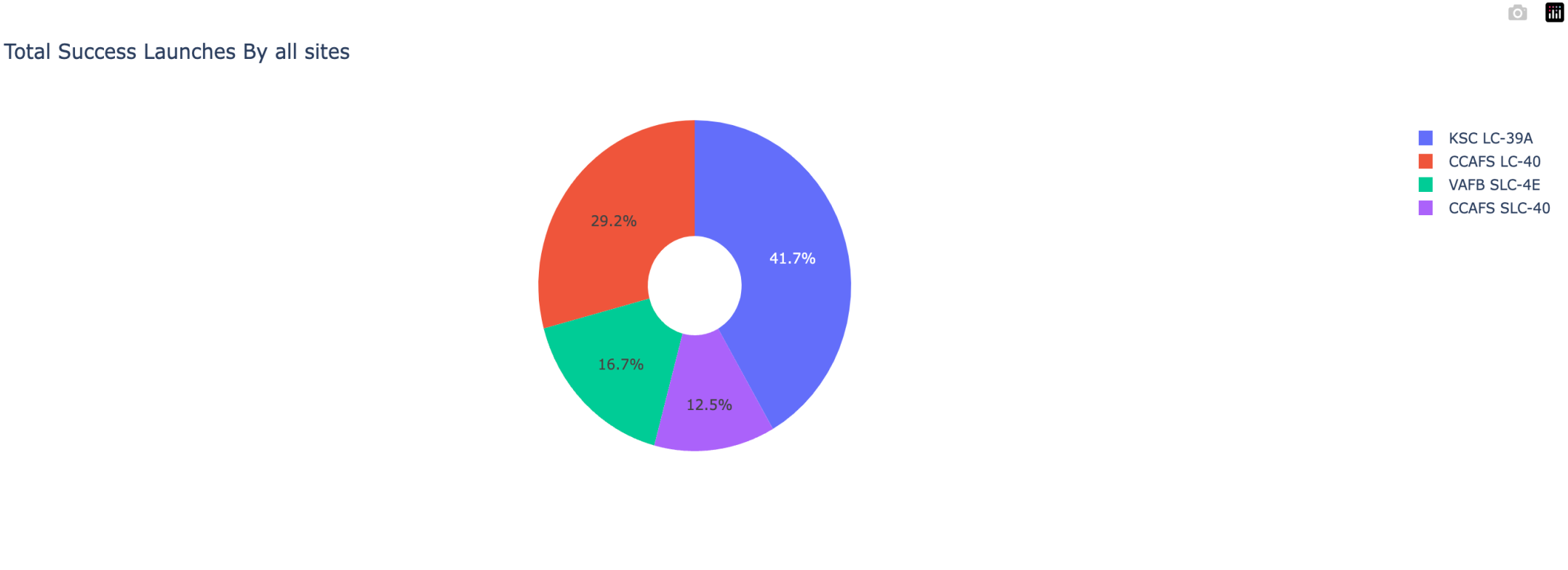




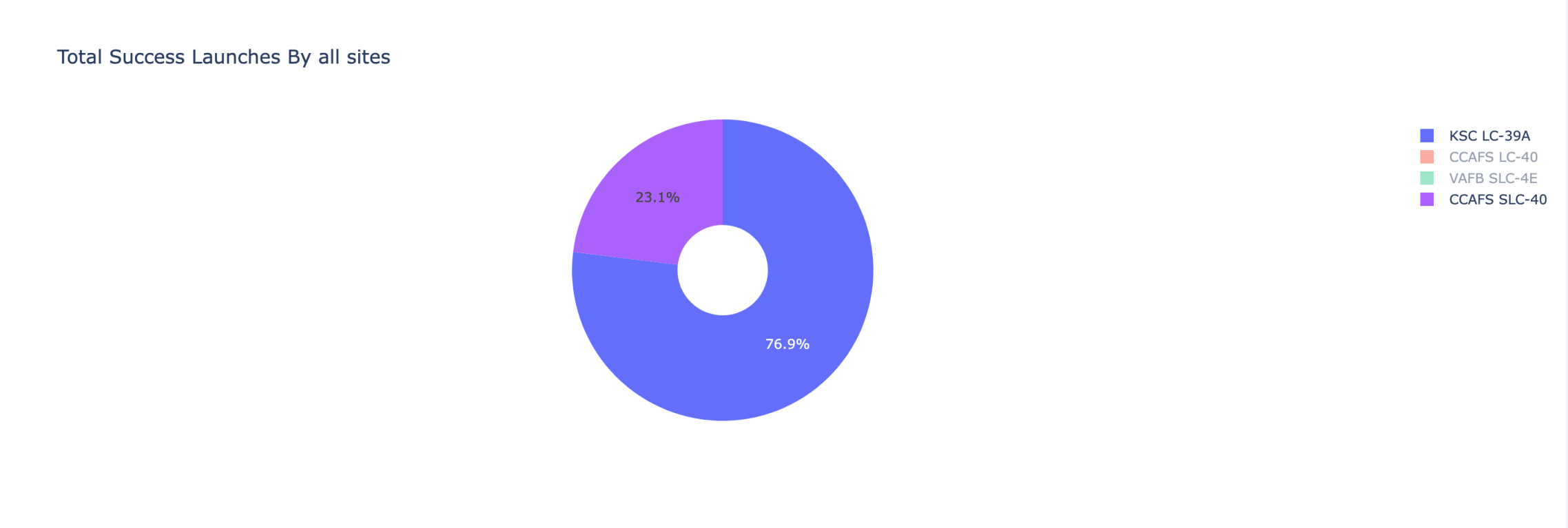



Section 4
Build a Dashboard with Plotly Dash

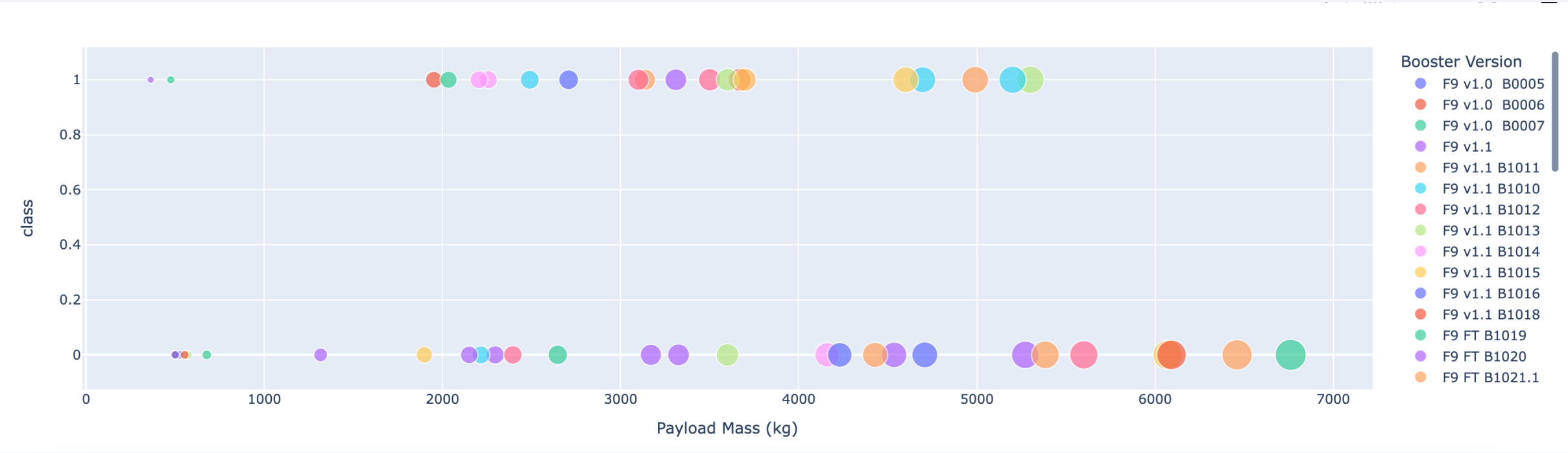
The success percentage by each sites



The highest launch-success ratio: KSC LC-39A



<Dashboard Screenshot 3>



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Find which model has the highest classification accuracy

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation

Conclusions

- Point 1
- Point 2
- Point 3
- Point 4
- …

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!